Introducing BASIC via the Overhead Projector

A Teacher's Guide to the use of EDUSYSTEM
overhead projector transparencies

VISUAL AIDS

The Visual Aids package consists of twenty-three (23) viewgraph transparencies designed for use with an overhead projector.

Planned as a teaching aid in the introduction of the BASIC language, each transparency is accompanied by a one page Teacher's Guide to the use of the viewgraph.

The transparencies and guides are:

- Terminal
  - 1A. DECwriter keyboard
  - 1B. Teletype keyboard
- BASIC order of operations
  - 2. addition, subtraction, multiplication, division, exponentiation
- BASIC statements
  - 3. statement numbers
  - 4. END
  - 5. LET
  - 6. PRINT
  - 7. INPUT
  - 8. READ-DATA
  - 9. GO TO
  - 10. FOR-NEXT
  - 11. IF-THEN
  - 12. DEF
  - 13. GOSUB-RETURN
  - 14. REMARK
  - 15. RANDOMIZE
  - 16. STOP
  - 17. ON-GO TO
- BASIC system commands
  - 18. SCRATCH, RUN and S
  - 19. LIST and DELETE
  - 20. EDIT
  - 21. TAPE and KEY
- BASIC functions
  - 22. SQR, ABS, INT, RND, LOG, EXP, SIN, LOS, TAN, ATN, SGN, TAB, FIX

A computer speaking BASIC is similar to a combination Superman and Albert Einstein - but only 4 years old.

Strong, obedient, extremely swift and infinitely patient, the computer will gladly perform as you direct it, but it only has a vocabulary of some 40 words.

Using these 40 words, you and your students can work wonders. You must, however, translate your higher-level thoughts into the simple, discrete vocabulary of the computer.

You will find the computer to be an eager and able student.

Introduce the computer's BASIC vocabulary to your sutdents a few words at a time. Allow them the short practice on the computer which will give them the mastery of the use of each new word.

A typical outline of the introduction of the BASIC vocabulary is:

LESSON	VIEWGRAPHS
1	14 or 18, and 2, 3, 4 and 18
2	5 and 6
3	19
4	7 and 9
5	8
6	10 and 14
7	11
8	12 and 22
9	13 and 16
10	15
11	17
12	20 and 21

The interval between these new introductions depends entirely on your students access to the computer and how much "hands on" experience they can acquire. You learn by doing. Your students should begin their hands-on work during Lesson 2.

One convention to be noted in all the following discussions is that ↵ denotes the "RETURN" key of the teleprinter and is used to terminate each line of typing.

Many of the transparancies (viewgraphs) contained in this kit show a successive interaction between an operator and the computer.

Cover all but the first line of typing with a sheet of paper, discuss this line and its meaning and then uncover each new line of print as your discussion advances.

Recognize one bright, blinding and sometimes upsetting fact of the instructional computer system.

YOU WILL NOT BE ABLE TO ANSWER ALL OF YOUR STUDENT'S QUESTIONS.

No one can.

Your role is that of initiator and guider. You do not want to answer all of their questions, but to agree "That is a dandy question -" and begin their searching "Why don't you check the Error section of the User's Guide for an idea?"

Even your slowest student will search and answer his own questions. He will gain more value from this one act of intellectual inquiry than a perfect knowledge of the Derivative would give him.

The computer will motivate and you will initiate and guide.

A final note - check the topics of the viewgraphs against your particular EduSystem User's Guide. A few language extensions such as EDIT are features of only some of 10 through 50.

Guide for Viewgraph 1A or 1B

The keyboard of an ASR-33 teletype or a DECwriter is not identical to that of an electric typewriter.

The ASR-33 teletype, in particular, was designed for use as a teletype machine - transmitting over wires to another teletype machine. So some of its keyboard functions are suited only to that use and are nonsensical in computer usage.

The elements you wish to point out to your students are:

the RETURN key	- BASIC will begin to do something only when you signify the conclusion of your typing by pressing the RETURN key. We will symbolize the RETURN key in any further discussion by ↩ .
the SHIFT key	- Discuss the special characters !" ( ) * + ↑ ← and how you hold down SHIFT while pressing O to get a ←, etc.
the 1 and L keys	- 1 has its own key and is not a lower case L. There are no lower case letters.
the Ø and O keys	- Ø is zero and O is the letter "oh". BASIC will not accept the letter O as a zero.
the CONTROL, ALT-MODE and LINE-FEED keys	- these special keys are used in a special way in advanced BASIC.

Guide for Viewgraph #2

BASIC is a language. As a language it has its unique grammar and rules of punctuation. Viewgraph #2 describes these rules.

Variables are either one letter, or one letter followed by one numeral. In the advanced use of BASIC, dimensioned (subscripted) variables may be used.

Examples of legal BASIC variables are:
X, Y, M, A1, B2, X(1), A(27), A1(3Ø), etc.

The operations of addition, subtraction, multiplication, division and exponentiation are represented in BASIC by +, -, *, / and ↑ respectively.

Particular note should be made of Algebra's assumed multiplication in expressions such as 2x, 15y, etc.

In BASIC, these would be 2*X and 15*Y.

The use of parentheses for grouping when in doubt is strongly encouraged. BASIC only requires that each left parenthesis be matched by a right parenthesis.

The order in which these operations are performed is depicted on the viewgraph.

Cover the viewgraph with a sheet of paper and uncover each new section as your discussion advances to that point.

Guide for Viewgraph #3

The BASIC compiler views any input in either of two modes, system (immediate) commands, and program (stored) commands.

BASIC assumes that anything preceded by a line number is to be stored for later execution. This allows you to enter an entire program of many statements before commanding execution.

BASIC assumes that any input that does not have a preceding number is a system command. BASIC will attempt to execute this command immediately.

For example, a student might wish to enter as the fifth line of his program the statement GOSUB 1ØØ. If he enters this without a line number the following printout appears:

```
GOSUB 1ØØ
WHAT?
```

The "WHAT?" is BASIC's response to what it thought was on immediate system command. Retyping this line as

```
5 GOSUB 1ØØ 2
```

enters the statement properly into the computer's memory.

Lastly, the order of execution of statements depends on the numerical order of their line numbers. As a consequence of this most BASIC users number their statements 1Ø, 2Ø, 3Ø, etc. so that if necessary they may later insert a line 15 between lines 1Ø and 2Ø.

Guide for Viewgraph #4

Every language has its grammar, or rules of usage. BASIC is a language and has such rules. One of the elementary grammatical rules is that every program must have an END statement as the highest numbered statement in the program, and this must be the only END statement in the program.

END performs two functions. It informs BASIC that execution may commence from the point of view of the entire program being in memory (essentially, someone told BASIC "That's all baby, there ain't no more."). If the computer works down through the statements and reaches the END statement, execution stops.

END marks the physical end of the program statements and halts execution if control reaches it.

Since this must be the highest numbered statement in the program and programs are quite often added to, many BASIC users number their END statements:

```
999 END
```

They then need not worry about expanding their program size and having to shift and reshift their "END" statement.

Guide for Viewgraph #5

LET is the assignment command. The general form is

<u>line number</u> LET <u>variable</u> = <u>expression</u>

Upon execution, BASIC will evaluate the expression on the right-hand side of the equals sign and store this value under the name given as the variable.

LET is optional and may be omitted. You may wish to require the use of LET for the first few weeks of programming since it reinforces the assignment or replacement function of LET.

For example: X = X+1 is a false Algebraic statement, but

```
1Ø LET X = X+1
```

is a perfectly legal and useful BASIC statement. This essentially says:

"Let X now be 1 greater than it was before."

LET is the computation statement of BASIC - this is where variables are defined and formulas evaluated. You will undoubtedly want to define a variable, say P, as 3.14159 rather than use the decimal approximation to Pi throughout your program.

Guide for Viewgraph #6

The PRINT statement is the output statement. This command causes the teleprinter to print as directed.

Any text enclosed within the double quote marks will be printed exactly as written.

```
10 PRINT "MY MUTHER LUVS ME."
```

will cause

```
MY MUTHER LUVS ME.
```

to be printed.

If the value of a variable, say X, has been previously defined in the program as 5, then

```
10 PRINT X
```

will cause

```
5
```

to be printed.

PRINT also will evaluate any expression and print the final value.

```
10 PRINT X↑2
```

will cause

```
25
```

to be printed.

Using a semi-colon between multiple variables in a PRINT statement will print the values of these variables with a blank (if positive) or a negative sign before the value and closely pack the printed values.

```
10 PRINT  X;3;-4
```

will cause

```
5 3-4
```

to be printed.

BASIC will space output into five print zones across the paper if commas are used as separators rather than the semi-colon.

```
10 PRINT X,3,-4
```

will cause

```
5          3          -4
```

to be printed.

The viewgraph gives a development of a program which not only outputs the square of X, but which also outputs the number being squared and the operation itself.

The output

```
5↑2 =  25
```

is much more descriptive and is generally to be desired over the output

```
25
```

which gives no information on what has occurred.

In general, use the semi-colon separator except when multiple columns of output are desired.

Note in the viewgraph development that retyping statement 20 replaces the old statement 20 with the new.

Use a sheet of paper to cover everything below the first "RUN" as you discuss the use of "PRINT" in the initial program and then uncover each new set of typing and output as your discussion advances.

Ask your students for their guesses as to what the output will be.

Guide for Viewgraph #7

The INPUT statement is the true interaction command of BASIC. This statement allows you to write a general algebraic program to solve some problem and not define the independent values until execution time.

Execution of the INPUT statement causes a question mark to be printed by the teleprinter and BASIC will then wait for the student to enter a value at the keyboard before execution continues.

If multiple variables are listed in the INPUT statement, the required number of values should be entered, separated by commas, at the keyboard.

Entering less than the required number of values will cause BASIC to continue printing a question mark and waiting for more input.

Too much input will generate an error message from BASIC.

Use a sheet of paper to cover the viewgraph below the first RUN and uncover each new section of typing and output as your discussion advances.

Guide for Viewgraph #8

The READ and DATA statements allow you to write a general solution program and insert new independent values prior to execution by retyping the DATA statements.

This function is similar to that of the INPUT statement but the use of READ and DATA allow a much higher throughput of programs in a batch system. The time delay inherent in a student responding to BASIC's wait on an INPUT command is avoided.

With an interactive multi-terminal system, INPUT is generally preferred by the user, but may prolong his stay at the terminal.

READ and DATA are of most value in an interactive system in defining the values of many variables.

For example:

```
10 LET A=1
20 LET B=2
30 LET C=3
40 LET D=4
50 LET E=5
```

may be replaced by:

```
10 READ A,B,C,D,E
20 DATA 1,2,3,4,5
```

Viewgraph #8 shows the functioning of the READ and DATA statements. READ A,B,C commands the computer, upon execution, to assign the first value of the DATA statement to A, the second to B and the third to C.

The first set of output verifies this as printing A,B and C does return the original DATA values of 3,5 and -1.

Retyping line 20 as 20 DATA 0,2,7 and commanding RUN repeats the program with the new DATA values.

Retyping line 2Ø again, but including four DATA values, and commanding RUN yields output consisting of just the first three values - no surprise, since the READ and PRINT statements only reference three values. The fourth DATA value is just ignored here.

Retyping line 2Ø as 2Ø DATA 1,2 and commanding RUN leads to an error message indicating that insufficient data was present. Line 1Ø asks for three values and line 2Ø only contains two.

Lastly, retyping line 2Ø as 2Ø DATA 1,2, BUM and commanding RUN will lead to another error message. This is due to line 1Ø having asked for three <u>numeric</u> values and line 2Ø containing as its third entry the non-numeric BUM.

Guide for Viewgraph #9

The GO TO statement has the general form

line number GO TO n

and upon execution causes control to be transfered to the statement numbered "n".

The example given in the viewgraph shows a general program that will continue to ask for a number and then produce its square. Line 3Ø is the GO TO which transfers control back up to line 1Ø.

Typing S will stop this "looping" and terminate the program run.

Changing line 3Ø to 3Ø GO TO 2Ø and commanding RUN will cause the program to accept one value and then loop around the PRINT statement of line 2Ø. The computer would continue to repeat this same printout until stopped - be it one minute, one day or one year.

The GO TO is the simplest and most used "branching" statement in BASIC.

Guide for Viewgraph #10

The FOR and NEXT statements define the beginning and end of a prescribed loop. "Prescribed" since the FOR statement not only defines how many times the loop will be executed, but also furnishes a "counter" variable which is incremented as the looping progresses. This counter is available to the programmer and may or may not be used in another statement as he sees fit.

For example if we wished to print MOTHER ten times we might write:

```
SCRATCH
1Ø FOR N = 1 TO 1Ø
2Ø PRINT "MOTHER"
3Ø NEXT N
999 END
RUN
```

and would get our ten MOTHER's.

If we wished to print out the integers from 1 through 2Ø and their squares we might write:

```
SCRATCH
1Ø FOR N = 1 TO 2Ø
2Ø PRINT N, N↑2
3Ø NEXT N
999 END
RUN
```

and we would get:

```
1          1
2          4
3          9
4          16
:          :
:          :
2Ø         4ØØ
READY
```

Just remember that FOR and NEXT must refer to the same variable to define a loop.

The STEP is a option included in the FOR statement when you wish your increment or step to be other than 1.

The viewgraph shows the optionality of the STEP in the third example.

Guide for Viewgraph #11

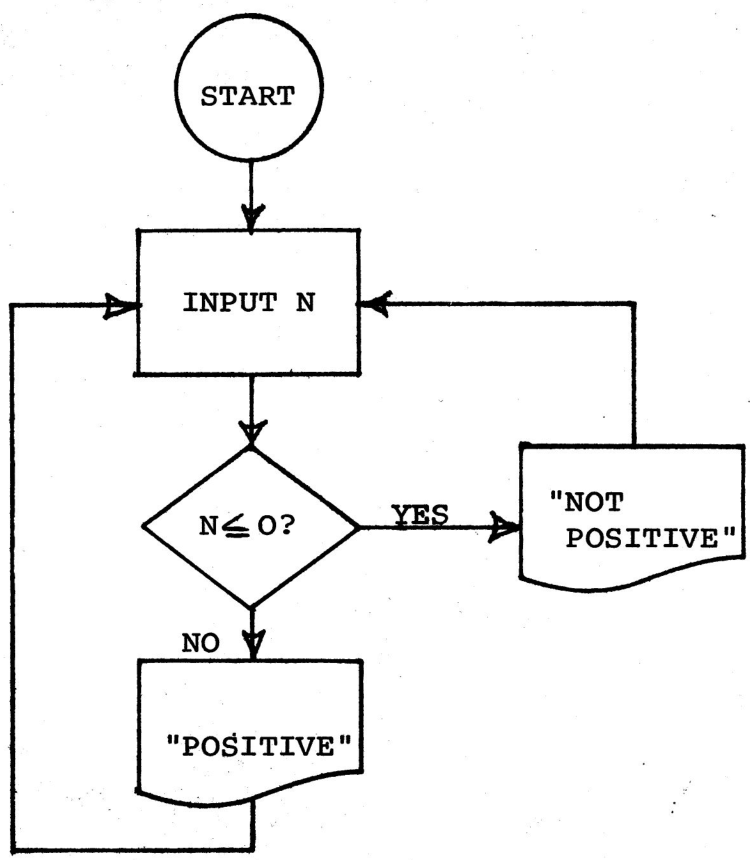
The IF-THEN (or IF-GO TO) statement is the decision maker statement.

IF-THEN matches the two given expressions with the relation given. If true, control passes to the statement whose line number follows THEN. If false, control "falls through" the IF-THEN to the next sequential statement.

The viewgraph illustrates a simple program which tests inputs and determines whether they are positive or not.

Walk through the program with your students and discuss the sequence of events following each new input.

The flow diagram which accompanies this program is:



Guide for Viewgraph #12

Programs written in Mathematics and the Sciences quite often have unwieldy expressions for functions which must be evaluated for many different values of the independent variable.

The DEF statement allows you to define a BASIC expression with its argument (substitutable independent variable) as your function and use just FNA(x) in the program rather than the entire expression.

The viewgraph illustrates how FNA(x) may be defined to be equivalent to $f(x) = x^2 + 3x-1\emptyset$. Then f(2), f(1), f(Ø) and f(-1) may be evaluated simply by inputting 2, 1, Ø and -1 as shown.

Relate FNA(x) to f(x).

Guide for Viewgraph #13

The GOSUB and RETURN statements are used to define a subroutine. The function of the subroutine is similar to the DEF statement. The subroutine, however, can perform much more work than the one calculation of a DEF.

As an example, consider a program where you would repeatedly use the distance formula, $d= \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$. Rather than write this long expression each time it appears, we can enter it just once - perhaps at the end of the program, and jump to it with a GOSUB and return with the RETURN. The program might look like:

```
2Ø INPUT x1, y1, x2, y2
3Ø GOSUB 2ØØ
 .
 .
7Ø INPUT x1, y1, x2, y2
8Ø GOSUB 2ØØ
 .
 .
13Ø INPUT x1, y1, x2, y2
14Ø GOSUB 2ØØ
 .
 .
19Ø STOP
2ØØ D = SQR ( (x1-x2)↑2+(y1-y2)↑2)

21Ø RETURN
999 END
```

Note that the statement just prior to the subroutine (statement 190) would normally be a STOP or GO TO as you would only want to enter the subroutine upon command.

The viewgraph illustrates a very simple use of GOSUB, but walking through the operation of the program with your students will serve to show its operation.

GOSUB <u>n</u> transfers control to the statement numbered n and control is returned to the statement immediately after the GOSUB when RETURN is encountered in the subroutine.

Notice the use of the semi-colon in the PRINT's of lines 2Ø and 7Ø to hold the carriage so that the ? of the INPUT appears on the same line.

Guide for Viewgraph #14

The REMARK statement is used by a programmer to clarify the operation of his program. If a section of his program performed a least - squares fit, then he might precede it with:

```
200 REM THIS SECTION PERFORMS A LEAST - SQUARES
210 REM FIT OF F(x) = A*X↑2+B TO THE DATA.
```

The purpose of the REMARK statement is to make your program readable, so that someone else can follow your steps and reasoning.

The REMARK statement is not executed.

Guide for Viewgraph #15

The use of the Monte-Carlo method in system design and exercise, the study of statistical phenomena and chance occurence may be simulated on the computer through the use of the RND function. This random number generator produces random decimal numbers between Ø and 1.

In the initial development of a program using RND, each running of the program will produce the same sequence of random numbers.

Once the program's logic and development have been verified, RANDOMIZE is inserted as the first statement.

Successive runnings of the program will then produce random sequences that are randomly started, i.e. the sequences are not the same.

Guide for Viewgraph #16

The STOP statement halts execution when control reaches it. Normally used to stop a program and prevent control from falling through to the next section, the STOP statement operates as a GO TO the END statement.

The viewgraph illustrates a simple use of the STOP statement.

Control of the given program can never reach the END statement and the STOP statement performs the halt of execution when a number other than 7 is entered.

Guide for Viewgraph #17

The ON - GO TO statement is a multiple branch statement the general form is:

ON expression GO TO n1, n2, n3, n4, ...

ON - GO TO evaluates "expression" and truncates it (drops off any decimal portion). If the remaining value is one (1) control is passed to the statement whose line number is first after GO TO. If the value is two (2), control is passed to the statement whose line number appears in the second place, etc.

Obviously, the truncated value of the expression must be at least one (1) and no larger than the number of line numbers following GO TO.

That is ON 5 GO TO 1Ø, 2Ø, 3Ø would result in an error state since there is no fifth line number in the GO TO list.

The example in the viewgraph illustrates this use of ON - GO TO.

Guide for Viewgraph #18

This viewgraph gives an illustration of the use of the system commands SCRATCH, RUN and S.

SCRATCH erases the working area in the computer's memory and gives you a clean piece of paper, so to speak, with which to work.

RUN begins execution of your program. If you have any grammatical errors in your use of BASIC, the appropriate error messages will appear when you type RUN.

Once your program is running, you may have the condition where you have intentionally, or unintentionally, gotten the computer into a loop, or perhaps waiting for input.

When you wish to stop execution of a running program, simply press the S key.

The computer will respond with READY.

Cover the viewgraph with a sheet of paper and uncover each section of typing and output as your discussion advances to that point.

The viewgraph illustrates a simple program with an infinite loop that requires the S to stop.

Notice that after the last SCRATCH, typing RUN only produces READY since the program has been erased by the SCRATCH command.

Walk through this and all the viewgraphs in a question-and-answer dialogue with your students - tracing the program steps.

Guide for Viewgraph #19

The LIST command will cause a printout of the computer's copy of your program. This is extremely useful in debugging (finding errors), because you quite often find that the computer's copy differs from your copy.

There are various reason's for this. If a student failed to type SCRATCH when he began writing his program, he will probably find some strange, left-over statements from the previous program nesting happily with his own.

He may have thought he typed PRINT when he really typed PTINT. His GO TO 3Ø may really be a GO TO 3ØØ, etc.

You can command LIST n, where n is a line number, and get a copy of a particular statement.

For example, if the computer were to answer your RUN command with ERROR IN LINE 5Ø, you can look at line 5Ø by typing LIST 5Ø. You should then be able to detect the error.

The DELETE command is used to delete or erase a statement or group of statements.

DELETE 1Ø will delete line 1Ø from your program, and DELETE 1Ø, 5Ø will delete lines 1Ø through 5Ø.

Cover the viewgraph with a sheet of paper and uncover each new section of typing and output as your discussion advances.

Guide for Viewgraph #20

The EDIT command is an extremely valuable labor saving extension to BASIC. EDIT allows you to correct text errors in a statement without having to retype the entire line.

To correct an error in a statement, type EDIT, the line number and the RETURN key. BASIC will then wait for a search character. Whatever character is next typed will cause BASIC to copy out the statement through the first occurence of that character. You then may:

Type	To Accomplish
CTRL/L (simultaneous depression of CONTROL and L keys)	continuation of copying through the next occurence of the search character
RUBOUT key or SHIFT/O keys (the ← character)	deletion of one character to the left for each depression
whatever characters you wish	corrections or insertions
CTRL/G	change of search character (more than one error in the line)
RETURN key	terminate editing, deleting any remaining text to the right
ALTMODE key	delete all characters to the left except the line number
LINE FEED key	terminate editing, saving the remaining characters to the right as presently written

The viewgraph depicts a sequence in which the original program uses the SQR function in line 2Ø, but has a missing right parenthesis.

When RUN is commanded, and error message is generated by BASIC indicating unmatched parentheses in line 2Ø.

Commanding LIST 2Ø outputs a copy of line 2Ø and the missing parenthesis is obvious.

The rest of the sequence illustrates the use of EDIT and primarily the CTRL/L option to move successively through the N's in the statement. Reaching the last N, we insert the missing parenthesis and terminate editing with the line feed.

We could have terminated with the return key, since there was no additional text to be saved, but students are too used to automatically pressing RETURN. In most cases, they will want to save the remainder of the line and should terminate with the line feed key.

Guide for Viewgraph #21

The reading and punching of paper tape requires the use of the TAPE and KEY commands.

The paper tape reader operates at a speed which is not identical to that of the teletype. If a paper tape is read in under normal operating conditions the incoming stream of characters are fed to the computer which retransmits them to the teletype buffer for echoing on the teletype. This stream fills the teletype buffer faster than the teletype can empty it and an overflow condition can occur.

To prevent this, the TAPE command is used. TAPE disenables the computer to teletype stream and the echoing of characters being read in does not occur.

The KEY command reenables the echoing of input on the teletype.

Typing TAPE and the return key prevents echoing. Pressing keys does not, then produce any echoing on the teletype.

Typing KEY and the return key does reenable echoing.

The viewgraph illustrates the procedure to follow in punching out, or reading in a paper tape.

Guide for Viewgraph #22

This viewgraph lists the functions provided by BASIC. These functions may be used in any BASIC expression.

Samples of their usage and the associated values are:

BASIC statement	Associated output
PRINT SQR (9)	3
PRINT ABS (-6)	6
PRINT INT (12.34567)	12
PRINT RND (Ø)	.2431684
PRINT LOG (4)	1.386294
PRINT EXP (1)	2.718282
PRINT SIN (3.14159)	2.668363E-6
PRINT COS (3.14159)	-1
PRINT TAN (3.14159)	-2.668363E-6
PRINT ATN (1)	.7853982
PRINT SGN (-23.45)	-1
PRINT FIX (-12.345)	-12

Note that INT (-12.345) = -13 while FIX (-12.345) = -12. INT is the Greatest Integer function.

RND generates a repeatable sequence of random numbers between Ø and 1. The sequence becomes non-repeating when the RANDOMIZE statement (viewgraph 15) is used in conjunction with RND.

Note that SIN (3.14159) and TAN (3.14159) generate values "sufficiently close to zero". This is due to both the approximation of pi and the round-off error in the routine which calculates these values. (2.668363E-6 = .000002668363 ≐ 0)

Finally, note that ATN (1) returns the radian approximation of 45 degrees.

Using degree measure for angles requires a conversion when calling SIN, COS or TAN:

```
10 PRINT "WHAT IS THE MEASURE OF YOUR ANGLE IN DEGREES";
20 INPUT D
30 R = D * 3.14159/180
40 PRINT "THE SINE IS"; SIN (R)
etc.
```

The inverse (180/3.14159) is used when converting from randians to degrees.